

PATENT APPLICATION

**AUTOMATED CREATION OF APPLICATION DATA
PATHS IN STORAGE AREA NETWORKS**

Inventor(s):

Christina Woody Mercier
140 Navigator Drive
Scotts Valley, CA 95066
a citizen of United States of America

Noel Christopher Melville
929 Windsor Street
Santa Cruz, CA 95062
a citizen of United States of America

Michael Carver Rieseberg
202 Woodlander Place
Scotts Valley, CA 95066
a citizen of United States of America

Nabin Acharya
424 Ribbonwood Avenue
San Jose, CA 95054
a citizen of Nepal

Truls Solheim Myklebust
219 Howes Drive
Los Gatos, CA 95032
a citizen of Norway

Assignee: InterSAN, Inc.
100 Enterprise Way, Suite C3
Scotts Valley, CA 95066

Entity: Small entity

AUTOMATED CREATION OF APPLICATION DATA PATHS IN STORAGE AREA NETWORKS

BACKGROUND OF THE INVENTION

The invention relates generally to the field of managing storage in a networked storage environment and, in particular, relates to the systems and methods for provisioning storage and paths, including devices in the path, to storage in storage networks.

Traditional computer storage architectures consist of dedicated storage devices connected to corresponding computer servers, an example of which is illustrated in Fig. 1. However, the direct attachment of storage to corresponding servers does not support sharing of the storage infrastructure by multiple servers. Thus, this architecture has major limitations, such as, for example, capacity and performance constraints, difficulty of management, reduced availability, and susceptibility to disasters.

The Storage Area Network (commonly known as "SAN") is a relatively new storage architecture that provides any-to-any connectivity between servers and storage at high (e.g., gigabit) speeds, allowing an enterprise to scale and manage its storage infrastructure independently of its server resources. For purposes of this invention, SAN is defined to include a storage network connecting a distributed and heterogeneous collection of compute servers, storage infrastructure, and special purpose, storage-service appliances. Fig. 2 illustrates a conventional SAN architecture 200 having computer servers 202, a switching network 204 and a storage infrastructure 206. Switching network 204 provides bi-directional coupling between servers 202 and storage infrastructure 206. Thus, SAN architecture 200 provides "any to any" connectivity between any servers 202 and a device of storage infrastructure 206, allowing heterogeneous collections of servers to share the same storage infrastructure. Information technology (IT) managers are implementing SAN technology to increase performance, consolidate and share storage resources, intermix best of class storage components, manage their storage in a centralized manner, and gain the availability features. In particular, the SAN architecture supports powerful new data management concepts in

support of backup, replication, and availability. For example, such “management” typically includes some or all of discovery, monitoring, topology creation, naming, persistence, data volume creation, switch security, data volume security, and authentication. A data volume is a single logical unit exported by a storage subsystem to a computer server. It may consist of many physical discs, portions of discs, or a data tape, but looks like one contiguous space to a server.

Unfortunately, conventional SAN management tools are typically server-based and have little or no inherent knowledge of the SAN special-purpose hardware and services such as replication services, snapshot engines, and virtualizers or SAN appliances. Conventional tools do not relate SAN storage and switch devices to applications and lines of business, and they don’t automate the many processes required to allocate storage to applications. As a result, the actions required to be taken by IT management to put SAN data management concepts into practice are very complex, and are becoming even more so as SAN infrastructures become more complex.

Today, most enterprises create data paths using a collection of device specific tools while referring to spreadsheets for SAN device information. Some enterprises have written scripts to address the complexity of storage provisioning in a SAN. These scripts have to be modified whenever new equipment is added to the SAN and executed by hand. In many cases they have selected one security mechanism rather than using all of the mechanisms to reduce complexity. In some cases, corporate databases have been created to maintain the settings of all the devices in the SAN although many organizations still use paper documents to keep track of current settings.

The conventional server-based SAN management tools typically provide only device-specific task management capability. Fig. 3 illustrates an example of the steps a SAN administrator would have to execute in order to provide a SAN attached server storage access using current un-automated, device-based SAN management tools.

At step 251, the SAN administrator physically verifies that the server is connected to the disc storage. At step 252, the SAN administrator finds a data volume of sufficient size. At step 253, the SAN administrator sets mapping of the selected data volume. At step 254, the SAN administrator selects one or more physical connections from the server to the storage subsystem with the data volume. At step 255, the SAN administrator verifies that the physical connection is good. At step 256, the SAN administrator configures the switch by setting security. Step 255 and 256 are repeated by

the SAN administrator for every switch used. Finally, at step 258, the SAN administrator configures the host bus adaptor (HBAs), within the servers.

Thus, as can be seen from the example of Fig. 3, the SAN administrator that administers a conventional SAN has a lot of storage infrastructure knowledge and
5 uses many different tools in order to effectively manage the SAN.

SUMMARY OF THE INVENTION

The present invention provides an efficient solution to the problem of implementation and management of a SAN. Preferred embodiments of the invention provide for both an apparatus and a method for automatic provisioning of storage to
10 servers within a SAN. An operator, rather than a highly trained storage and switching expert, is able to perform automated provisioning which results in the creation of a data path between a server and data. A preferred embodiment of the present invention discovers and saves details of the SAN architecture, including, for example, server configurations, processes executable on specific servers and association of the processes
15 with the server, devices and configurations of the switching network, and devices and configurations of the storage architecture. Devices, as used herein, are defined to include, for example, disc storage subsystems, tape storage subsystems, storage switches, storage routers, SAN appliances, and other storage devices such as solid state disc, etc.

Not only is static information determined, but dynamic information and
20 state information as well. In the preferred embodiment, a DataPath Engine is provided which initiates, controls and monitors the discovering, saving, using, configuring, recommending and reporting associated with the preferred embodiment. The DataPath Engine calculates the optimal data path based upon the rules or policies specified and information learned about the SAN, including policies and rules defined in preconfigured
25 templates for interaction with the DataPath Engine. As used herein, the term template is defined to include, for example, a list of defined rules and policies which define the storage characteristics and data path characteristics that must be used by the DataPath Engine for selection of a data path. The template is created in advance by an administrator using a graphical wizard, for example.

30 A preferred embodiment of the present invention is directed towards a method of creating a data path for a process executing on a server coupled to a storage area network (SAN). The method includes parameterizing a set of attributes for a desired

data path between the process and a device of the SAN; and constructing the data path that provides the set of attributes. For purposes of this application, the term attributes includes details about data volumes, security settings, performance settings, and other device and policy settings., and parameterizing is defined to include defaults selected by the system to help the administrator make better choices when creating a template which reflects data path policy and rules., with parameterizing attributes referring to an abstraction of the configuration, implementation and creation steps to identify the desired end product without necessarily specifying implementation details.

An alternate preferred embodiment of the present invention provides for a method of configuring a SAN. The method includes discovering, by use of a DataPath Engine coupled to the SAN, processes that are operable on a server coupled to the SAN; discovering, by use of the Data Path Engine coupled to the SAN, devices that are included in the SAN; responding, by use of the DataPath Engine coupled to the SAN, to a data path construction request from a user by providing the user with an interface to accept a set of attributes for a desired data path for one of the discovered processes; and constructing, by use of the DataPath Engine coupled to the SAN, the data path that provides the set of attributes.

In one aspect, the method of a preferred embodiment includes discovery of SAN device details; accepting policy and input regarding the type and size of data volume and path desired; and finding candidate data paths and volumes that meet the policy. The term policy and rules are important parts of the preferred embodiment of the present invention. Policy is defined to include actions which the DataPath Engine will take based upon events in the SAN and the term rules is defined to include characteristics of data volumes and data paths that the DataPath Engine must use to select candidate data paths for the application.

The data path may contain multiple channels or threads. A thread is a logical relationship representing a physical path between the server on which the application is resident and all of the devices, connections, ports and security settings in between. Further, for purposes of this application, threads are defined to include one or more of, depending upon the needs of the embodiment, application id, server id, HBA port id, HBA id, HBA security settings, switch port ids, switch security settings, storage subsystem port id, data volume id, data volume security settings, SAN appliance port id, SAN appliance settings. These relationships include, but are not limited to, the data

volume; the storage subsystem the volume resides on; all ports and connections; switches; and SAN appliances and other hardware in the data path; the server with the Host Bus Adapter (HBA) where the application resides; and all applicable device settings. The data path selection is based upon policies such as, number of threads, number of separate storage switch fabrics that the threads must go through, level of security desired and actions to take based upon security problems detected, performance characteristics and cost characteristics desired. Data paths are created from SAN devices automatically discovered by the DataPath Engine (Applications, Servers, HBAs, Switches, Fabrics, Storage Subsystems, Routers, Data Volumes, Tape drives, Connections, Data Volume security, etc.). The data path can have multiple threads to the same data volume and span physical locations and multiple switched fabrics.

In another aspect, a preferred embodiment of the present invention is a method including an apparatus for selection and creation of the optimal data path among the candidate data paths. Pathing methodologies within the DataPath Engine use discovered information about the SAN such as device uptime information, performance information, cost information, and load. Device uptime information is defined to include, for example, the collection and persistence of data about when a device or connection or service is in service and available versus unavailable. Performance information is defined to include the collection and persistence of data about how each device is moving data from its location to the next and the resulting rate of data that the application is experiencing. Cost information is defined to include the correlation of cost information with actual devices used within a data path taking into consideration the percentage of the device used by the data path, when multiple data paths share the same devices. Cost in this case is a calculation of the infrastructure used. Many customers purchase expensive hardware in hopes of achieving high levels of availability. A calculation of the cost (switch ports used, HBA ports used, # gigabytes used) for each path may be provided for customers to analyze their cost for performance and availability achieved, as well as used as a "rule" when creating data paths.

Best practices information is also factored in as appropriate for the discovered devices, such as fan out ratios and switch fabric architecture which impacts performance once a new data path is added to the infrastructure.

Implementations of preferred embodiments of the invention provide for one or more of the following:

An operator with no storage or switch training uses a wizard on a graphical viewer to provision storage. Provisioning storage is defined to include creating a data path for a software application on a SAN attached server to a new or existing data volume.

5 Data path creation rules and policies are specified in a pre-created template. The template is expandable to include new rules as they are defined. A policy or template is created by a SAN administrator to meet the availability and performance needs of a software application on a SAN attached server. At storage provisioning time the proper template is automatically selected for the application. In some cases, 10 predefined templates are provided and do not require creation by the SAN administrator. In some instances, for example, this is appropriate for standard implementations of common processes/applications.

Data path thread selection logic uses pathing methodologies that take into consideration the learned state and usage of the SAN.

15 Once the selected data path is approved by the operator, the DataPath Engine automatically configures SAN devices for data path creation across multiple devices, networks and locations.

Implementations of automated storage provisioning include but are not limited to, creation of data paths for an application, discovery of pre-existing data paths, reconfiguration of data paths, movement of data paths between asynchronous replications, and tuning of data paths based upon data collected about the SAN's performance and uptime. Advantages of the invention include the ability for a small number of operators to manage large, complex and distributed storage networks. They do not require detailed knowledge of storage networking devices or extensive training. Manual procedures and 20 policies are automated for a huge time gain and reduction of personnel. Pathing methodologies calculate the best data paths rather than relying on experts or operator memory to select the optimal path during setup. Complex storage networking hardware and services can be added to storage networks and quickly incorporated into new or existing data paths.

30 Incorporating new devices into new or existing data paths is preferably done by adding new "rules" to the template. For example, a special purpose storage replication system is added to the SAN. A rule would be added to the DataPath Engine that allows a template to be created which selects a data path with the specified

replication systems. New paths would be created using this template. Existing paths would be changed by data re-pathing. Re-pathing would allow an operator to select an alternate path to the existing volume or a replica of the volume (in another location) using a selected template. This capability, then, will support switching between replications as well as incorporating new devices into existing paths by changing to a new template (policy and rules). This also supports a change in performance characteristics and path optimization via a template with different settings than the original one used.

In an alternate preferred embodiment, the DataPath Engine stores the parameterization data or accesses a store of parameterization data used in the specification of existing data paths (including policies/templates/rules) used in guiding the generation of each existing data path. Periodically (automatically or operator initiated), the DataPath Engine reruns the pathing methodologies based upon the stored parameters to determine whether a new optimal data path exists. Depending upon specific embodiments, the data path may be changed automatically or the user may be requested to authorize the use of the new data path.

Automated storage provisioning is a powerful system for enterprises with extensive storage networks to reduce their personnel requirements and better utilize their storage infrastructure. As used herein, the term automatic means that all the underlying SAN infrastructure and settings are configured by the DataPath Engine without administrator intervention based solely on a request specifying an application, data volume size and template.. The above description refers to the construction of a data path.

Further understanding of the nature and advantages of the invention may be realized by reference to the remaining portions of the Specification and Drawings. In the drawings, similarly numbered items represent the same or functionally equivalent structures.

BRIEF DESCRIPTION OF THE DRAWINGS

The invention will be described with respect to particular embodiments thereof, and reference will be made to the drawings in which:

- Fig. 1 illustrates the "direct attach" storage architecture of the prior art;
- Fig. 2 illustrates the Storage Area Network (SAN) architecture of today;

Fig. 3 illustrates the steps, which a SAN administrator must perform for the prior art of manually provisioning storage within the SAN architecture shown in Fig. 2;

Fig. 4 is a block diagram illustrating a preferred embodiment for a system that performs automated storage provisioning;

Fig. 5 illustrates the steps the DataPath Engine and an operator to perform automated storage provisioning within the SAN architecture shown in Fig. 4;

Fig. 6 illustrates rules or policies that can be specified by a SAN administrator which the DataPath Engine uses to select data volumes and data paths for an application;

Fig. 7 is a layout showing connections for Fig. 7A and 7B which together illustrate a flowchart of a pathing methodology used by DataPath Engine to select the optimal data path through a complex SAN infrastructure;

Fig. 8 is an illustration of a graphical user interface 800 for, in the preferred embodiment, provisioning a data path;

Fig. 9 is an illustration of interface 800 after the operator has selected 'Next' from interface 800 configuration illustrated in Fig. 8;

Fig. 10 is an illustration of interface 800 after the operator has selected 'Next' from interface 800 configuration illustrated in Fig. 9; and

Fig. 11 is illustrates an alternate preferred embodiment for creation of a data path. In this case, a data path is desired to an existing data volume, already being used by another server.

DESCRIPTION OF THE SPECIFIC EMBODIMENTS

Fig. 4 illustrates an embodiment in accordance with the invention in which DataPath Engine 302 is provided as part of (or closely coupled to, depending upon a particular embodiment) the SAN infrastructure; switching network 204, SAN attached servers 202, and storage infrastructure 206. DataPath Engine 302 operates in response to high-level SAN management commands entered by the operator via a Management Station 304 using a graphical user interface. The DataPath Engine is preferably operated as part of a general purpose computer server running any of several types of operating systems as well-known art; with one or more processing units; connected to the Wide Area Network (WAN) and switching network 204; and having attached a persistent

repository 306. The DataPath Engine executes best with world-wide connectivity to SAN infrastructure via WAN and fibre channel connections and can be clustered with multiple computer servers for high availability and work load balancing. The management station is preferably operated as part of a specially programmed general purpose computer, such as a PC, workstation, mini-computer or other computing platform running any of several types of operating systems as well-known in the art. As well known, these platforms include one or more processing units for executing instructions obtained from memory (volatile or nonvolatile) and typically under control from signals provided through an I/O system. A display adapter typically maps memory contents of page frames for display on a monitor. Instructions and data for the executing instructions (program) and other data may be in the memory systems, which also includes removable media such as a portable magnetic media (e.g., floppy disk). The processing unit is able to communicate with other processing systems (e.g. the servers in the server architecture, devices in the switching/routing architecture, or devices in the storage architecture, typically by employing conventional networking protocols such as TCP/IP for example. These systems are well known to persons of ordinary skill in the art and will not be further explained herein.

DataPath Engine 302 is coupled to switch network 204 and WAN 208 and obtains knowledge of the identity and behavior of the specific devices in storage infrastructure 206 and applications on the SAN attached servers 202. This information is embedded, incorporated or otherwise associated within DataPath Engine 302, saved to a persistent repository, and obtained either by automated discovery or through manual configuration. Automated discovery includes polling and broadcasting across the WAN and switch network for devices to initially find devices; to find new devices added to the SAN infrastructure, and to continually collect status on the devices. The configuration and use of DataPath Engine 302 allows device management specifics to be hidden from the operator as it handles all the details of individual device management.

Fig. 5 illustrates procedures for automated storage provisioning by an operator using DataPath Engine 302. (This is in contrast to the discussion above concerning Fig. 3, which illustrates path allocation using conventional SAN management.) At step 351, DataPath Engine 302 automatically discovers SAN device configuration and status. At step 352, the operator selects a create data path command from a graphical interface along with policy. The embodiment of the command type is

“Create a Virtual Private DataPath”. A “Virtual Private DataPath (VPD)” is the term representing a data path coupled with a data volume meeting the template rules, and monitored constantly in order to implement policies specified. At step 353, DataPath Engine 302 selects the best candidate data volumes that meets the policy criteria. At step 354 the optimal candidate data paths are displayed for operator selection and the data path is set up based upon the operator’s action. A data path represents the number of threads created between the application server and the data volume.

In this preferred embodiment, the pathing methodologies prioritize certain selection requirements over other possible candidates. In other embodiments, different selection requirements may produce different optimal paths. Optimal refers to a best fit of available resources to parameterized attributes based upon applicable prioritization conditions. In other embodiments and under different conditions, the recommended or optimal data path could be different, so best is used in a relative sense as opposed to an absolute sense.

Thus, it can be seen that an operator using the system illustrated in Fig. 4 can manage the SAN without having detailed knowledge of the SAN devices and quickly provision storage with a two-step command.

Fig. 6 illustrates rules that can be implemented as policy for data path selection. The policy includes, but is not limited to: Storage Type, 451, Number of Threads or Channels, 452, Number of Fabrics, 453, Bandwidth, 454, Data Volume Security, 455, Data Volume Action, 456, Fabric Security, 457, Fabric Action, 458, HBA Security, 459, HBA Action, 460, Data Volume Size, 461, DataVolume Pooling, 462, Data Volume Characteristics, 463, and Data Path Status, 464.

Fig. 7 illustrates a pathing methodology process 550 setting forth a preferred embodiment of how DataPath Engine 302 in Fig. 4 selects an optimal path. Process 550 first reads a data path policy 551, obtains a current SAN state 552, gets an application name and volume size desired from an operator 553, and initiates a search for data paths from each of the server’s HBA ports 554.

The data path search 554 loops through each server HBA port 555, using the following steps: checks whether a port is used by any other data path 566; when already used, then gives the port a weighting 557, 558; when not used, checks to see whether used by another thread of this data path 559; and when not then gives the port the best ranking 560; and adds the port to candidate list 561.

Once all candidate ports have been identified, DataPath Engine 302 determines whether the number of HBA ports is equal to or greater than the number of threads specified in the policy 562, and when not, it fails (Step 563). Otherwise it searches for storage subsystems that match the policy requirements 564.

5 The data volume search loops through each storage subsystem 565, and determines whether there are suitable data volumes 566. The DataPath Engine 302 loops through each data volume 567, and when the size is acceptable 568, and the data volume is accessible 569, it gives the data volume a ranking 571. When no data volumes are found 572, then it fails 573. All data volumes inaccessible from the server HBA are then discarded 574.

10 Next DataPath Engine 302 ranks each switch fabric 575, and computes the shortest data path 576. For each HBA port on the server 577, it finds the shortest route through the fabric to the data volume 578, and ranks the data path 579. Finally, DataPath Engine 302 calculates and sorts data paths by their ranking 580, 581, 582.

15 Fig. 8 is an illustration of a graphical user interface 800 for, in the preferred embodiment, parameterizing a set of attributes for a desired data path. Interface 800 is coupled to DataPath Engine 302 and may be part of management station 304. The preferred embodiment uses a simple set of data entry screens 800 and those shown in Figs. 9-11, to guide an operator in the definition and creation of an optimal data path (the path from the application on the server to the data volume on a storage subsystem). Interface 800 includes data based upon the context from which the operator requested to create the optimal data path. For example, when the operator has the owner view for “Trading Floor” currently opened, the owner information of interface 800 contains “Trading Floor.” The operator is able to override any field. In some cases, interface 800 is prepopulated (or preloaded) with policy information or template information.

20 Fig. 8 through Fig. 10 illustrate a data path creation scenario for a preferred embodiment. Fig. 8 describes an interface 800 that is populated with information to serve as an example. Interface 800 includes a data path template “GoldService” identifying and specifying rules about data path creation that hides the complexity and details of the storage and switching network architectures from the operator. The template includes information such as the number of threads, the security levels, default data volume size, etc. The provisioning can be done on behalf of an application executing on one server or a cluster of servers. In this example the operator is

creating a data path from the application, Oracle, which is executing on a cluster consisting of three servers (DBserver1, DBserver2, DBserver3). The size requested is 72GB. The owner of the data path is the Trading Floor. Included in interface 800 is the concept of data path owner, which for purposes of this application, is defined to include a name of a customer or line of business within an enterprise and information about that organization such as a contact person, their email address for automatic notification, their address, applications which are associated with the owner and default templates that are associated with the owner.

Fig. 9 is an illustration of interface 800 after the operator has selected 'Next' from interface 800 configuration illustrated in Fig. 8, if "automatically select best VPD" is not checked. After the operator selects Next, the DataPath Engine 302 searches the SAN and switching infrastructure for candidate data volumes and the best data paths from each server to candidate data volumes. The best data path candidates will be selected using the pathing methodologies specified in Fig. 7. Fig. 9 illustrates that interface 800 now displays a set of data path candidate sets, including best data path candidate (determined as the candidate data path matching the rules and policies most closely) to each server, and alternates. The operator selects an alternate set or continues to the Next screen which automatically selects the default best candidate data paths.

Fig. 10 an illustration of interface 800 after the operator has selected 'Next' from interface 800 configuration illustrated in Fig. 9. Once the operator selects Next from the previous interface 800 configuration, interface 800 provides the operator with an opportunity to name the data path and to provide a description. After the DataPath Engine 302 creates the identified data path, the DataPath Engine 302 notifies the operator.

Fig. 11 illustrates an alternate preferred embodiment for creation of a data path. In this case, a data path is desired to an existing data volume, already being used by another server. This is a common case with clusters of servers that must all access the same data for a shared application. In this case interface 800 is configured as shown in Fig. 11 and is used in place of interface 800 illustrated in Fig. 8. Other than the data volume preexist, the subsequent steps in the data path creation process do not have to be changed.

It should be noted that the embodiments described here may be implemented in hardware, software, firmware or some combination thereof. While

particular embodiments have been described, the scope of the invention is not to be limited to any particular embodiment. Rather, the scope of the invention is to be determined from the claims.

